**package** Question26;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** LargestMultipleOfThree {

**public** **static** **int** find(**int**[] array) {

/\*\* sort in ascending order \*\*/

Arrays.*sort*(array);

/\*\* To hold those digits such that digit % 3 == 0 \*\*/

Queue<Integer> queue0 = **new** LinkedList<>();

/\*\* To hold those digits such that digit % 3 == 1 \*\*/

Queue<Integer> queue1 = **new** LinkedList<>();

/\*\* To hold those digits such that digit % 3 == 2 \*\*/

Queue<Integer> queue2 = **new** LinkedList<>();

**int** sum = 0;

**for** (**int** i = 0; i < array.length; i++) {

sum += array[i];

**switch** (array[i] % 3) {

**case** 0:

queue0.add(array[i]);

**break**;

**case** 1:

queue1.add(array[i]);

**break**;

**default**:

queue2.add(array[i]);

**break**;

}

}

**switch** (sum % 3) {

**case** 1:

**if** (!queue1.isEmpty()) {

*dequeueX*(queue1, 1);

} **else** {

*dequeueX*(queue2, 2);

}

**break**;

**case** 2:

**if** (!queue2.isEmpty()) {

*dequeueX*(queue2, 1);

} **else** {

*dequeueX*(queue1, 2);

}

**break**;

}

/\*\* Merge all digits from did\*\*/

Integer[] digits = *mergeQueue*(queue0, queue1, queue2);

/\*\* Sort digit in reverse order \*\*/

Arrays.*sort*(digits, Collections.*reverseOrder*());

**return** *formNumber*(digits);

}

**private** **static** **int** formNumber(Integer[] digits) {

StringBuilder sb = **new** StringBuilder(digits.length);

**for** (**int** i = 0; i < digits.length; i++) {

sb.append(digits[i]);

}

**return** Integer.*valueOf*(sb.toString());

}

**private** **static** Integer[] mergeQueue(Queue<Integer> queue0, Queue<Integer> queue1, Queue<Integer> queue2) {

Integer[] array = **new** Integer[queue0.size() + queue1.size() + queue2.size()];

**int** i = 0;

**while** (!queue0.isEmpty()) {

array[i++] = queue0.remove();

}

**while** (!queue1.isEmpty()) {

array[i++] = queue1.remove();

}

**while** (!queue2.isEmpty()) {

array[i++] = queue2.remove();

}

**return** array;

}

**private** **static** **void** dequeueX(Queue<Integer> queue, **int** numberOfTimes) {

**try** {

**for** (**int** i = 0; i < numberOfTimes; i++) {

queue.remove();

}

} **catch** (Exception e) {

**throw** **new** RuntimeException("Number not possible ");

}

}

**public** **static** **void** main(String[] args) {

**int**[] digits = {8, 1, 7, 6, 0};

System.***out***.println(LargestMultipleOfThree.*find*(digits));

}

}